Locating Sensors to Observe Network Arc Flows: Exact and Heuristic Approaches.

L. Bianco[*], C. Cerrone[†], R.Cerulli[†], M. Gentili[‡]

Abstract

The problem of *optimally* locating sensors on a traffic network to monitor flows has been object of growing interest in the past few years, due to its relevance in the field of traffic management and control. Sensors are often located in a network in order to observe and record traffic flows on arcs and/or nodes. Given traffic levels on arcs within range or covered by the sensors, traffic levels on unobserved portions of a network can then be computed. In this paper, the problem of identifying a sensor configuration of minimal size that would permit traffic on any unobserved arcs to be exactly inferred is discussed. The problem being addressed, which is referred to in the literature as the Sensor Location Problem (SLP), is known to be NP-complete, and the existing studies about the problem analyze some polynomial cases and present local search heuristics to solve it. In this paper we further extend the study of the problem by providing a mathematical formulation that up now has been still missing in the literature and present an exact branch and bound approach, based on a binary branching rule, that embeds the existing heuristics to obtain bounds on the solution value. Moreover, we apply a genetic approach to find good quality solutions. Extended computational results show the effectiveness of the proposed approaches in solving medium-large instances.

Keywords: Sensor Location, Traffic Network, Branch and Bound, Genetic Algorithm.

[*]Dipartimento di Ingegneria dell'Impresa. Università di Roma "Tor Vergata". Via del Politecnico 1, 00133 Roma, Italy. bianco@disp.uniroma2.it

[†]Dipartimento di Matematica. Università di Salerno. Via Giovanni Paolo II n. 132 , 84084 Fisciano (Sa), Italy. carminecerrone@gmail.it, raffaele@unisa.it, mgentili@unisa.it

[‡]Corresponding Author. Tel. +39 089 963444, Fax +39 089 96 3303

1 Introduction

Monitoring flows on the network is an important topic in the field of traffic management and control. The continuous growth in the demand for private transportation in large urban areas is the cause of severe congestion, pollution, time loss in traffic jams and a deterioration in the quality of life. Monitoring flows on a network allows traffic managers to control and manage these problematic situations. Even though communication technologies for monitoring traffic networks in real time, via sensors and video cameras, are currently available, in most cases we have a very large network which is monitored only in small part. In this context, partial information on traffic flows (obtained, for example, by located sensors on the network), is often used to estimate flows in the network that are not directly observed. However, recent studies in the literature show that by properly locating sensors on the network it is possible to exactly compute, under the assumption of error free data, the entire set of unobserved flows. In this context it is of paramount importance to design optimal strategies to determine sensor locations, and the problem of locating sensors on the network has been object of growing interest in the past few years. Problems in this class are differentiated according to the type of sensors that need to be located (counting sensors, path-ID sensors, vehicle-ID sensors or a combination of them) and flows of interest (origin/destination flow volumes, arc flow volumes, route flow volumes, or a combination of them). *Counting sensors* can be considered all those types of sensors that are able to count vehicle on a lane(s) of a road (for example conventional inductive loop sensors). They can be located on an arc (a vertex) of the network and count the number of vehicles on the arc (on the arcs incident to the vertex) during a particular time interval. *Path-ID sensors* are assumed to be devices that, when located on an arc of a network, can measure the flow volume of each route to which that arc belongs. This is the class of sensors that de-code active transmission provided by tagged vehicles, for example, freight information from trucks or path/schedule information from buses. *Vehicle-ID sensors* are sensors through which a vehicle can be univocally identified while traveling on the network. Licence plate readers or Automatic Vehicle Identification (AVI) readers are examples of sensors that belong to this class.

Following the classification defined in the recent surveys by Gentili and Mirchandani [14]-[15], two classes of problems can be identified: (i) locating sensors to fully observe flow values (either arc flows, route flows or OD flows) on the network (Sensor Location Flow-Observability Problems), and (ii) locating sensors to estimate flows (either arc flows, route flows or OD flows) on the network (Sensor Location Flow-Estimation Problems). This dichotomy is derived from the observation that the location of sensors on a network (either on the vertices or on the arcs) can be translated into a system of linear equations where the set of variables corresponds to the unknown flows and the set of equations comes from the deployed sensors. When the resulting system has a unique solution, we say the system is fully observable, and therefore all the flows involved in the system are known (that is, they are *observable*). The resulting

location problem consists of determining the optimum deployment of sensors on the network that results in an observable system (Sensor Location Flow-Observability Problems). On the other hand, when the system is underspecified, it admits an infinite number of solutions. The related location problem consists of determining how to optimally deploy sensors on the network so that the derived flow estimates are as good as possible. Generally, underspecified systems arise when one is interested in determining origin-destination flow volumes by locating counting sensors on the arcs of the network. This problem has been extensively studied in the literature (see for example, Chootinan et al. [10], Elhert et al. [12], Yang et al. [18] Kim et al. [19], Lam and Lo [20], Yang and Zhou [23]). On the other hand, observability problems arise, for example, when either path-ID sensors or vehicle-ID sensors need to be located on the arcs of the network to determine route flow volumes (Gentili and Mirchandani [15]-[16], Castillo et al. [6], [7], [8], Cerrone et al. [9]).
In this paper we focus on the observability problem arising when counting sensors are located on the vertices of the network and arc flow volumes need to be computed. Specifically, we are interested in locating the minimum number of counting sensors on the vertices of a network to compute arc flow volumes on the whole network (we refer to this problem as the *Sensor Location Problem* [*SLP*]).

There is an extensive literature related to facility location on networks and, in particular, to the Flow Interception Problems. The problem addressed in this paper could seem similar to this well known class of problems; however, there is a huge difference. In flow interception problems a set of facilities is to be located on the network (generally on the arcs of the network) to intercept flows such that a given function of the flow is optimized (e.g., total intercepted flow is maximized [17] or total risk reduction is maximized [13]). Any subset of arcs (or vertices) that is selected is feasible for the problem. The only similarity between the SLP and this class of problems is the fact that once a facility (a sensor) is located on the network, a flow is intercepted. The main issue that makes the SLP unique and different from the flow interception problems is that the location of the facility has to be such that from the directly intercepted flows all the non-directly intercepted flows on the remaining arcs of the network can be computed. Hence the network becomes fully observable in terms of arc flows.

The Sensor Location Problem was formally stated by Bianco et al. [5], where two heuristics giving lower and upper bounds on the solution value were presented, and a necessary condition for feasibility was stated. Successively, Bianco et al. [4] developed a combinatorial analysis of the problem and studied its computational complexity considering different special cases. Moreover, some graph classes, where the problem is polynomially solvable, were also presented. Morrison and Martonosi [21] and Morrison et al. [22] addressed the problem on bi-directed trees, giving a necessary and sufficient condition for a subset of vertices to be a feasible solution of the problem. They also defined a matrix reduction procedure to test feasibility of a subset of vertices. Confessore et al. [11] further studied the problem and presented new heuristic algorithms and approximation algorithms. The quality of the solutions provided by the existing algorithms is evaluated by comparison with data-dependent approximation bounds. Indeed, for the problem being considered there does not exist any exact approach providing (at least for small instances) the exact solution value, nor does there exist a mathematical formulation.

In this paper we further extend the study of the problem by (i) developing and testing an exact approach, based on a branch and bound scheme, to optimally solve the problem, (ii) developing and testing a genetic solution algorithm to get near-optimal solutions, and, (iii) providing a mathematical formulation of the problem based on the concept of MB-paths introduced in [4]. The mathematical formulation provided is a flow based formulation whose optimal solution provides a lower bound on the optimum solution value of SLP. We point out here that this is the first attempt to formulate this problem whose mathematical model was still missing in the literature. We tested our approaches on the set of benchmark instances existing in the literature, on new instances we developed to extend the test cases for the problem and on a real world network. Our results show the efficiency of the proposed approaches in solving medium-large instances.

The paper is organized as follows. Section 2 describes the problem being addressed and details the results in the existing literature that will be used in the present study. Our mathematical formulation is given in Section 3. Section 4.1 describes our genetic approach, while our Branch and Bound algorithm is described in Section 4.2. Computational results are discussed in Section 5 and Section 6. Conclusions and further research are the subject of Section 7.

2 Problem description and existing results

In this section we describe in detail the problem being addressed. For this entire discussion we assume that data are error-free. Given a directed graph $G = (N, A)$, where N is the set of n vertices and A is the set of m arcs, the flow on each arc contains subflows that are generated and/or absorbed from different origin/destination pairs. Among the set of vertices, we say v is an *Origin/Destination (OD)* vertex if the flow is generated and/or absorbed by it. The set of OD vertices of G is denoted by $B \subseteq N$. An OD vertex can be either the origin or the destination of flows or both, and it can also be used as a transfer node. Each OD vertex can send flow to or receive flow from any other OD vertex.

For each vertex v that is not an OD one, the flow conservation constraints hold:

$$\sum_{w \in FS(v)} f_{v,w} - \sum_{w \in BS(v)} f_{w,v} = 0 \tag{1}$$

where $FS(v)$ and $BS(v)$ are the outgoing and incoming arcs of vertex v, respectively, and $f_{v,w}$ is the flow volume on arc (v, w).
For each OD vertex $v \in B$, we have the following flow conservation constraint:

$$\sum_{w \in FS(v)} f_{v,w} - \sum_{w \in BS(v)} f_{w,v} = S_v \tag{2}$$

where $S_v \neq 0$ is the *balancing flow* at v, that is, a source or a sink flow so that (2) holds. If split ratios[*] associated with the arcs of the network are also known we could define additional relationship involving the flows. Indeed, by using split ratios, we can express the total outgoing flow $F(v)$ as a function of the flow volume of any outgoing arc. Formally, for each $v \in N$ and each outgoing arc (v, w), by using split ratios $p_{v,w}$ at arc (v, w), we have:

$$f_{v,w} = F(v) \cdot p_{v,w} \tag{3}$$

From (3) and considering any other outgoing arc of v, say (v, z), we obtain:

$$f_{v,z} = \frac{f_{v,w}}{p_{v,w}} \cdot p_{v,z} \tag{4}$$

To better understand the definitions introduced before, refer to the example network in Figure 1 with 6 vertices and 14 arcs. Table 1 gives the unknown arc flows on each arc together with known split ratios associated with the arcs of the network. Hence, for example, flow on arc $(2, 6)$ is equal to 10 (i.e., $f_{2,6} = 10$) and flow on arc $(4, 5)$ is equal to 5 (i.e., $f_{4,5} = 5$). The net flow at vertices 1,2,3, and 6 is equal to zero; vertices 4 and 5 are OD vertices with balancing flows equal to $S_4 = 10$ and $S_5 = -10$, respectively. We have, for example, $F(1) = 30$, $f_{1,3} = F(1) \cdot p_{1,3}$ where $p_{1,3} = \frac{2}{3}$, and, $f_{1,2} = F(1) \cdot p_{1,2}$ where $p_{1,2} = \frac{1}{3}$. When locating a counting sensor on a vertex we assume that we know the flow volumes on all the arcs incident to the vertex. Suppose we locate a counting sensor on vertex 1, that is, flows $f_{1,2} = 10$, $f_{2,1} = 20$ $f_{1,3} = 20$, $f_{3,1} = 10$ are directly monitored and become known. We show now how these monitored flows, together with the knowledge of the split ratios, allow us to compute the flows on all the remaining arcs of the network. By using the split ratio equation (4), it is possible to define a system of linear equations where (i) a single unknown variable, representing an outgoing arc flow, is associated with every vertex, except the monitored vertex and its adjacent ones, and (ii) equations correspond to

[*]The split ratio associated with the outgoing arc (v, w) specifies the fraction $0 \leq p_{v,w} \leq 1$ of the outgoing flow $F(v)$ that leaves vertex v the arc (v, w).

Arc	Arc Flow	Split Ratio
$(1,2)$	10	$\frac{1}{3}$
$(1,3)$	20	$\frac{2}{3}$
$(2,1)$	20	$\frac{4}{9}$
$(2,4)$	15	$\frac{3}{9}$
$(2,6)$	10	$\frac{2}{9}$
$(3,1)$	10	$\frac{1}{3}$
$(3,5)$	20	$\frac{2}{3}$
$(4,2)$	30	$\frac{6}{7}$
$(4,5)$	5	$\frac{1}{7}$
$(5,3)$	10	$\frac{2}{5}$
$(5,4)$	10	$\frac{2}{5}$
$(5,6)$	5	$\frac{1}{5}$
$(6,2)$	5	$\frac{1}{3}$
$(6,5)$	10	$\frac{2}{3}$

Table 1: *Arc flows and split ratios associated with the arcs of Figure 1.*

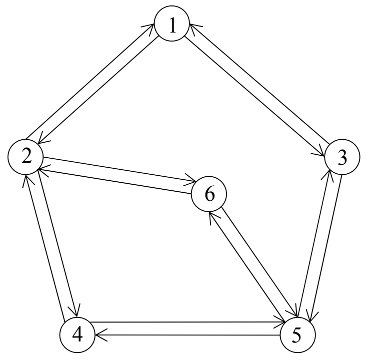

Figure 1: *An example network with six vertices.*

flow conservation constraints relative to every vertex of the network. Additional variables S_v, denoting the balancing flows of OD vertices, are also included in the linear system. Note that the flow conservation equation corresponding to the measured vertex 1 does not contain any unknown flow, and thus it can be omitted from the system. When vertex 1 is monitored, we obtain the following system corresponding to the flow conservation constraints of the network:

$$\begin{array}{lccccc}
\textit{vertex } 2 & (f_{2,1} + f_{2,6} + f_{2,4}) & - & (f_{1,2} + s_{6,5} \cdot \frac{p_{6,2}}{p_{6,5}} + s_{4,2}) & = & 0 \\
\textit{vertex } 3 & (f_{3,1} + f_{3,5}) & - & (f_{1,3} + s_{5,4} \cdot \frac{p_{5,3}}{p_{5,4}}) & = & 0 \\
\textit{vertex } 4 & (s_{4,2} + s_{4,2} \cdot \frac{p_{4,5}}{p_{4,2}}) & - & (s_{5,4} + f_{2,4}) & = & S_4 \\
\textit{vertex } 5 & (s_{5,4} + s_{5,4} \cdot \frac{p_{5,3}}{p_{5,4}} + s_{5,4} \cdot \frac{p_{5,6}}{p_{5,4}}) & - & (f_{3,5} + s_{4,2} \cdot \frac{p_{4,5}}{p_{4,2}} + s_{6,5}) & = & S_5 \\
\textit{vertex } 6 & (s_{6,5} \cdot \frac{p_{6,2}}{p_{6,5}} + s_{6,5}) & - & (f_{2,6} + s_{5,4} \cdot \frac{p_{5,6}}{p_{5,4}}) & = & 0
\end{array} \qquad (5)$$

where: the unknown variables are $s_{4,2}$, $s_{5,4}$, $s_{6,5}$, S_4 and S_5; flows $f_{1,2}$, $f_{2,1}$, $f_{1,3}$, $f_{3,1}$ are directly monitored; and flows $f_{2,4}$, $f_{2,6}$ and $f_{3,5}$ are obtained using split ratios. The remaining unknown flows are obtained using equation (1). In this system, the number of variables is equal to the number of equations, the underlying coefficient matrix has full rank; therefore, the system has a unique solution. Of course, different systems of equations are associated with different sets of vertices. We refer to a subset M of vertices whose associated modified flow conservation system has a unique solution as a *monitoring set*. Each monitoring set then represents a set of vertices where sensors should be installed on the network so that all the arc flows can be inferred. Due to the high cost of these devices, minimizing the total cost is equivalent to minimizing the total number of devices to be installed. Hence, the arising location problem consists of finding a monitoring subset $M \subseteq N$ of minimum size (the *Sensor Location Problem - SLP*).

The SLP is NP-complete as proved in [4]. The next section presents some known results that will be useful in the subsequent sections of the paper. In particular, we will give a brief description of some

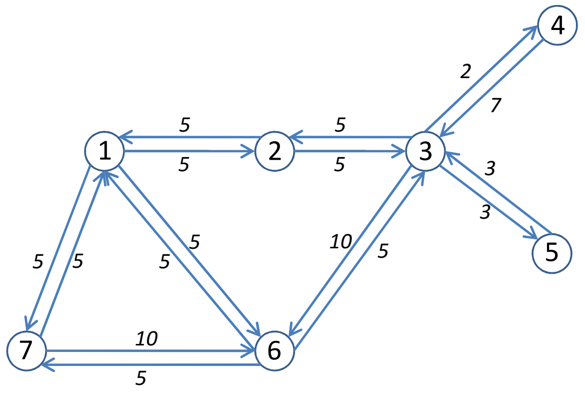

Figure 2: *An example network with seven vertices.*

combinatorial properties of the problem, its formal definition and an existing algorithm for finding a lower bound on the solution value.

2.1 Known results about SLP

Before presenting the needed results, we introduce some additional definitions.
Given a subset of vertices $M \subseteq N$, we denote by $Adj(M)$ the subset of vertices that are adjacent to one or more of the vertices in M. Consider the network of Figure 2, with 7 vertices and 16 arcs. If $M = \{1\}$, then the set of adjacent vertices is $Adj(M) = \{2, 6, 7\}$. A *combined cut* $C_M \subseteq A$ associated with M is the subset of arcs in G induced by $M \cup Adj(M)$. Hence, in our example, the combined cut associated with $M = \{1\}$ is $C_M = \{(1, 2), (2, 1), (1, 6), (6, 1), (1, 7), (7, 1), (7, 6), (6, 7)\}$. Let us denote by $G \setminus A'$ the graph obtained from G after deleting the subset of arcs $A' \subseteq A$, and, when no confusion may arise, by $G \setminus N'$ the graph obtained from G after deleting the subset of vertices $N' \subseteq N$. Let $G_i^D = (N_i^D, A_i^D)$ denote the i-th connected component of $G \setminus D$ (sometimes denoted simply $G_i = (N_i, A_i)$). Let B_i and $Adj(M)_i$ be the set of OD vertices and adjacent vertices in the generic i-th component, respectively. Consider again the network in Figure 2 and assume there are two OD vertices on the network, say vertex 4 and vertex 7, that is, $B = \{4, 7\}$. The graph $G \backslash C_M$ obtained from G after deleting the combined cut C_M associated with $M = \{1\}$ has three connected components: the connected component G_1 with vertex set $N_1 = \{2, 3, 4, 5, 6\}$, a degenerate component G_2 whose vertex set is $N_2 = \{7\}$ and a degenerate component G_3 with vertex set $N_3 = \{1\}$. Note that each vertex $v \in M$ will form a degenerate component of $G \backslash C_M$, and all the remaining components of the graph do not contain (by construction) vertices of M. Let us refer to the degenerate components containing vertices of M as the *monitored* components and to all the other connected components as the *not monitored* components. Hence, in our example, component G_1 is a not monitored component and contains two vertices of $Adj(M)$, which are $Adj(M)_1 = \{2, 6\}$, and one OD

vertex: that is $B_1 = \{4\}$. Component G_2 is a not monitored component such that $Adj(M)_2 = B_2 = \{7\}$, and component G_3 is a monitored component with $Adj(M)_3 = B_3 = \emptyset$. If G is a bi-directed graph and M is a monitoring set, then each not monitored connected component G_i in $G \backslash C_M$ is such that the total number of OD vertices in G_i is not greater than the total number of adjacent vertices in the same component as stated in the following proposition by Bianco et al. (for details of proof see [5]).

Proposition 1 *Let G be a bi-directed graph. If $M \subseteq N$ is a monitoring set, then*

$$|B_i| \leq |Adj(M)_i| \quad \forall i = 1, 2, \ldots, q. \tag{6}$$

where q is the number of not monitored connected components of $G \setminus C_M$.

Consider again our network example in Figure 2 with $M = \{1\}$ and $B = \{4, 7\}$. Clearly, the non monitored connected components of $G \backslash C_M$ satisfy the necessary condition stated in Proposition 1. Note that, if vertices 2 and 6 were also OD vertices, then condition (6) would not be satisfied; indeed the connected component G_1 would contain 2 adjacent vertices and 3 OD vertices. Then, in this case, we can say $M = \{1\}$ is not a monitoring set, that is, the associated system does not admit a unique solution.

By the definition of connected component, every pair of vertices in the component is connected by a path. Therefore, condition (6) can be restated in terms of paths connecting OD vertices and adjacent vertices, as stated in the following proposition by Bianco et al. [4].

Proposition 2 *Let G be a bi-directed graph. If $M \subseteq N$ is a monitoring set, then there exist in G, $|B \setminus M|$ number of MB-paths distinct in the origin and in the destination.*

where an MB-path is a path P starting from a vertex in $Adj(M)$ and ending at an OD vertex such that:

(C1) P does not contain two consecutive vertices that belong to $Adj(M)$;

(C2) P does not contain vertices of M.

Referring again to Figure 2 where $M = \{1\}$ and $B = \{4, 7\}$, we could define the $|B \setminus M| = 2$ MB-paths: $P_1 = \{2, 3, 4\}$ and $P_2 = \{7\}$. Again note that, if we assume vertices 2 and 6 are also OD vertices, then we cannot find a sufficient number of MB-paths that are distinct in the origin and in the destination. It is worth noting that condition (6) (and hence the condition stated in Proposition 2), is necessary but not sufficient. Indeed, assume on the network in Figure 2 that vertices 4 and 5 are OD vertices, that is, $B = \{4, 5\}$. Assume we locate a counting sensor on vertex 6, that is $M = \{6\}$. Then the combined cut is $C_M = \{(6, 7), (7, 6), (6, 1), (1, 6), (1, 7), (7, 1), (6, 3), (3, 6)\}$, for which we have three connected components: the not monitored component G_1 with vertex set $N_1 = \{1, 2, 3, 4, 5\}$, the not monitored component

```
Step 0. (Initialization) $M = \emptyset$;
Step 1. Sort $N$, namely, arrange the vertices in $N$ according to decreasing value of $\sigma(v)$.
Step 2. Choose the head vertex in $N$, remove it from $N$ and put it into $M$.
Step 3. If $\sum_{v \in M} \sigma(v) \geq |B|$ , then STOP. Else, goto step 2.
```

Table 2: Algorithm LB.

G_2 with vertex set $N_2 = \{7\}$ and the monitored component G_3 with $N_3 = \{6\}$. Component G_2 trivially satisfies Proposition 1. In G_1, the total number of adjacent vertices is $|Adj(M)_2| = 2$ and the total number of OD vertices is $|B_2| = 2$; hence, also this connected component satisfies condition (6). Moreover, we can define $|B \setminus M| = 2$ MB-paths, $P_1 = \{1, 2, 3, 4\}$ and $P_2 = \{3, 5\}$, that are distinct in the origin and in the destination. Hence, Proposition 2 is satisfied too. However, it can be verified that the system associated with the set $M = \{6\}$ does not admit a unique solution, and therefore M is not a monitoring set.

The necessary condition stated in Proposition 1 and Proposition 2 can be used to define a simple greedy procedure to compute a lower bound on the optimum solution of the problem. In particular, the LB algorithm presented in [5] is a greedy algorithm that selects vertices in $v \in N$ according to a decreasing order with respect to their degree $\delta(v)$, chooses the one with the highest number of adjacent vertices (plus one if the vertex is an OD vertex) and places a sensor on it. Vertices are selected until condition (6) is satisfied on the entire graph without taking into account the connected components induced by the combined cut C_M. The pseudocode of the algorithm is given in Table 2 where parameter $\sigma(v)$ is equal to $\delta(v) + 1$ if v is an OD vertex and is equal to $\delta(v)$ otherwise.
For example, for the graph in Figure 2 with $B = \{4, 5\}$, the lower bound obtained by applying the LB algorithm is 1, which is, in this simple case, also the optimum solution.
The minimum cardinality subset of vertices satisfying condition (6) can be obtained by solving the mathematical formulation provided in the next section. It is a flow commodity formulation that looks for a subset of vertices M satisfying Proposition 2.

3 Mathematical Formulation

The mathematical formulation we propose is a single commodity flow formulation where each OD vertex requires a unit of flow that originates from a vertex in $Adj(M)$. In particular, we consider three different sets of variables: (i) binary variables $z_v = 1$ if a sensor is located at vertex v and $z_v = 0$ otherwise; (ii) binary variables $y_v = 1$ if a sensor is either located on vertex v or if vertex v is adjacent to a vertex where a sensor is located and $y_v = 0$ otherwise; (iii) flow variables $x_{v,w}$ associated with each arc of the graph. The provided mathematical formulation is the following:

$$
\begin{aligned}
& min \sum_{v \in N} z_v && && (7)\\
& s.t.: && && \\
z_v + \sum_{w \in FS(v)} z_w &\geq y_v && \forall\, v\, \in N && (8)\\
z_v + \sum_{w \in FS(v)} z_w &\leq n y_v && \forall\, v\, \in N && (9)\\
\sum_{w \in FS(v)} x_{v,w} - \sum_{w \in BS(v)} x_{w,v} &= y_v - 1 && \forall\, v \in B && (10)\\
\sum_{w \in FS(v)} x_{v,w} - \sum_{w \in BS(v)} x_{w,v} &\leq y_v && \forall\, v \in N \backslash B && (11)\\
\sum_{w \in FS(v)} x_{v,w} - \sum_{w \in BS(v)} x_{w,v} &\geq 0 && \forall\, v \in N \backslash B && (12)\\
2n - n(y_v + y_w) &\geq x_{v,w} && \forall\, (v,w) \in A && (13)\\
z_v &\in \{0,1\} && \forall\, v \in N && (14)\\
y_v &\in \{0,1\} && \forall\, v \in N && (15)\\
x_{v,w} &\geq 0 && \forall\, (v,w) \in A && (16)
\end{aligned}
$$

The objective function (7) seeks to minimize the total number of selected vertices. Constraints (8)-(9) ensure that variables y_v exactly define vertices that are selected or that are adjacent vertices: y_v is equal to 1 if either vertex v is selected or at least one of its neighbors is selected. In particular, when neither v is selected (that is, $z_v = 0$) nor one of its adjacent vertices (that is, $\sum_{w \in FS(v)} z_w = 0$), constraints (8) become active and force $y_v = 0$. On the other hand, constraints (9) become active and impose $y_v = 1$ when the left side is greater than or equal to 1; these can happen into two cases: either when $z_v = 1$ and $\sum_{w \in FS(v)} z_w \geq 0$ or when $z_v = 0$ and $\sum_{w \in FS(v)} z_w \geq 1$. Constraints (10) - (13) ensure the definition of MB-paths connecting vertices in $Adj(M)$ with vertices in B. In particular, each MB-path is defined by sending one unit of flow from the adjacent vertex where it starts to the OD vertex where it ends; hence, one unit of flow must be originated from any adjacent vertex that constitutes the beginning of an MB-path and must be absorbed by an OD vertex that constitutes the end of an MB-path, all the other vertices in the graph must have a zero balancing flow. For any OD vertex v (i.e., $v \in B$), the net flow can be equal either to 0 or to -1 according to the role of the vertex. Indeed, the following cases may occur for any $v \in B$: (i) vertex v is selected, (ii) vertex v is an adjacent vertex, (iii) vertex v is neither selected nor it is an adjacent vertex. In the first case, vertex v is part of a degenerate monitored connected component and it does not constitute the end of any MB-path, hence, it does not absorb any flow unit and its net flow must be equal to zero; in the second case, being $v \in B \cap Adj(M)$, vertex v can be considered both the starting point and the end point of an MB-path, and hence, it both generates and absorbs one unit of flow and the net flow results equal to zero; finally, in the third case, vertex v must be the end of an MB-path, and hence, it must absorb one flow unit and the net flow in this case must be equal to -1. These three cases are handled through constraints (10) that ensure the flow balance at any vertex $v \in B$ is equal to zero flow units if either $v \in M$ or $v \in Adj(M)$ (that is, $y_v = 1$), and the flow balance is equal to -1 otherwise (that is, $y_v = 0$). Flow conservation constraints for the other vertices are expressed by (11) and (12). If vertex $v \in Adj(M)$ (i.e., $y_v = 1$), then it may or may not be the origin of an MB-path, and hence, it may or may not originate one unit of flow: constraints (11) and (12) ensure the net flow on the vertex is less than or equal to one; on the other hand, if $v \notin Adj(M)$ (i.e., $y_v = 0$), then it is a transfer vertex and constraints (11) and (12) force the net flow on the vertex to be equal to zero. Finally, constraints (13) ensure that each MB-path is such that it contains neither (i) arcs incident to selected vertices (condition C1) nor (ii) arcs connecting two vertices both belonging to $Adj(M)$ (condition C2). Indeed, the flow variable $x_{v,w}$ associated with arc (v, w) is forced to be zero whenever either (i) $v \in M$ and $w \in Adj(M)$ or (ii) $v \in Adj(M)$ and $w \in Adj(M)$.

We outline that flow variables $x_{v,w}$ are used here to define the MB-paths required to satisfy the condition stated in Proposition 2. These variables have nothing to do with actual trips on the network which are represented by variables $s_{v,w}$ and S_v in the modified conservation system associated with the subset of selected vertices. The actual trips on the network can be obtained by solving this modified conservation system when it results to be full rank.

4 Our solution approaches

In this section we describe our proposed solution approaches for the problem. In particular, we developed a Branch and Bound approach and a genetic algorithm to solve SLP. The feasibility test implemented for both of our solution approaches takes into account the combinatorial rule suggested by condition (6). In particular, given a subset of vertices M, two cases can occur:

- If property (6) is not satisfied, then we do not need any other verification: M is not a monitoring set.
- Otherwise, if property (6) is satisfied, then the current solution *might* be a monitoring set; hence, we build the associated modified flow conservation system and check whether it admits a unique solution or not.

The methodology adopted for building the modified flow conservation system associated with a subset of vertices is the one proposed in [21] and [22] and it is reported in the appendix.

4.1 The Genetic Algorithm

A genetic algorithm is a randomized search optimization heuristic, which is based on the biological process of natural selection and can be applied to a variety of combinatorial optimization problems. Genetic algorithms mimic the evolutionary process by processing a population of solutions simultaneously. Starting with solutions that are typically created randomly, better ones are more likely to be chosen for recombination with others to form new solutions. Thus new solutions inherit good parts from old solutions. In addition to recombining solutions, new solutions may be formed through mutating or randomly changing old solutions. The process is repeated until stopping criteria are met. In this way, good solutions or also optimal ones could be reached.The main components of our algorithm are described below while the pseudocode is given in table 3.

Solution Representation
The generic chromosome used to represent a solution is a vector with a number of components equal to $|B|$ (that is, the total number of OD vertices for the instance being considered, which is also an upper bound on the optimal solution value). The generic component of the chromosome (the gene) can assume an integer value in the interval $[1, n]$ where $n = |N|$ is the total number of vertices in the graph.
Fitness Function
The fitness function associated with each chromosome is based on the total number of different genes in the chromosome (i.e. the total number of located sensors $|M|$) and the level of infeasibility of such a choice (dependent on the number of not monitored connected components G_i, induced by the combined cut C_M, where property (6) is not satisfied). In particular, let us suppose the selected sensors define the set M and let q be the number of not monitored connected components induced by the combined cut C_M where the necessary condition (6) is not satisfied. In this case, the fitness function value associated with M is:

$$f(M) = \begin{cases} |M| & \text{if } q = 0 \text{ and } M \text{ is a monitoring set} \\ |M| + |B| & \text{if } q = 0 \text{ and } M \text{ is a not monitoring set} \\ |M| + |B| * (h+1) & \text{otherwise} \end{cases}$$

where

$$h = \sum_{i=1}^{q} (|B_i| - |Adj(M)_i|)$$

For example, let us consider the graph in Figure 2 with 7 vertices, and suppose there are four OD vertices, i.e., $B = \{2, 4, 6, 7\}$. We would have a chromosome with four elements, and each element could take a value in the interval $[1, 7]$. Chromosome $CH_1 = [2, 4, 6, 1]$ represents a feasible solution (that is, a monitoring set) where vertices 1, 2, 4 and 6 are selected; the corresponding value of the fitness function

```
Step 1. Randomly generate $p$ chromosomes to serve as the initial population; set $i = 0$.

Step 2. Build generation $i + 1$ from generation $i$ as below:
        - Apply the crossover and mutation operator on the population $P_i$
        to generate $p$ chromosomes in population $P_{i+1}$.
        - Eliminate the $y\%$ worst chromosomes and substitute them with the
        best chromosomes of population $P_i$

Step 3. Repeat Step 2 until $i \leq MaxIteration$.

Step 4. Output the best solution obtained so far.
```

Table 3: Genetic Algorithm: Overview.

is $f(\{1, 2, 4, 6\}) = 4$. Chromosome $CH_2 = [1, 1, 1, 1]$ represents an infeasible solution where vertex 1 is selected and the corresponding fitness function value is equal to $f(\{1\}) = 1 + 4 * (1 + 1) = 9$.

Crossover Operator

The crossover operator generates two new chromosomes (children) starting from two chromosomes (parents) belonging to the actual population. That is, two parent chromosomes, say CH_i and CH_j, are randomly selected from the actual population. The two children CH_{q_1} and CH_{q_2} are generated as follows. A position $h \in [1, |B|]$ is randomly generated, and the genes from position h up to position $h + k$ (where k is a previously specified parameter) are exchanged between the two parent chromosomes to generate the two children. For example, in the above example with parents chromosomes $CH_1 = [2, 4, 6, 1]$ and $CH_2 = [2, 2, 1, 5]$, by letting $h = 2$ and $k = 1$ the two generated children would be $CH_3 = [2, 2, 1, 1]$ and $CH_4 = [2, 4, 6, 5]$.

Mutation Operator

The mutation operator makes small random changes in the individual of the population, which provides genetic diversity and enables the GA to search a broader space for the problem. For each newly generated child, we adopted a fixed mutation scheme that changes a single randomly selected gene with a new randomly generated value.

Population Size and Initial Population

We chose to generate populations whose size is fixed and equal to a parameter p. All the p chromosomes that constitute the initial population are randomly generated.

Population Reproduction Operation

Let P_i be the actual population. Then the new population P_{i+1} is generated as follows: (i) generate p chromosomes by applying the crossover and mutation operator, (ii) replace the $y\%$ worst chromosomes with the best chromosomes of population P_i.

In our implementation of the algorithm, we set the size of the population to $p = 100$. The percentage y in the reproduction operation is equal to 50%. The algorithm stops when the computational time is greater than n seconds.

4.2 The Branch and Bound Approach

In this section we describe our branch and bound algorithm to find an exact solution to the problem. The search tree is a binary search tree formed by at most n levels, each corresponding to a vertex of the graph. Therefore, at level k of the search tree, we have 2^k decision nodes since the branch rule consists of either choosing or not choosing vertex k in the graph to be part of the solution.
The LB algorithm (described in Section 2.1) is used inside the Branch and Bound routine to find at each node of the search tree a lower bound on the solution value. In particular, such a procedure has been modified (see Section 4.2.1) to solve the following constrained version of the SLP:

Constrained Sensor Location Problem ($CSLP$)
INSTANCE: A bi-directed graph $G = (N, A)$, a subset $B \subseteq N$, two disjoint subsets of vertices $I \subseteq N$ and $O \subseteq N$ and a positive integer $K \leq |B|$.
QUESTION: Is there a monitoring set $M \subseteq N$ of vertices with $|M| \leq K$, such that $I \subseteq M$ and $M \cap O = \emptyset$?

A CSLP is associated with each decision node of the search tree, say node h, and two sets, say I_h and O_h, represent the partial solution built so far. The subset I_h contains vertices of the graph that have already been selected to be part of the partial solution, and the subset O_h contains vertices of the graph that have been designated as not being part of the partial solution, according to the choices made in the path connecting the root node of the search tree to node h. In particular, let us define a set of binary variables x_i associated with each vertex $i \in N$ of the graph whose values are such that $x_i = 1$ if vertex i is selected and $x_i = 0$ otherwise. Consider now, for example, the partial binary search tree in Figure 3 associated with the graph of Figure 2. The first level of the tree corresponds to vertex 1 of the graph which can be chosen to be part of the solution ($x_1 = 1$) or not ($x_1 = 0$); the second level of the tree corresponds to vertex 2 of the graph and so on. An instance of the CSLP is associated with each node of the tree according to the values of the variables fixed so far. For example, the two disjoint sets $I_7 = \{1, 4\}$ and $O_7 = \{2, 3\}$ are associated with node 7 of the search tree.

The branch and bound scheme consists of a sequence of *forward moves* and *backtracking* moves. A forward move selects a vertex of the graph to be part of the solution (by setting the corresponding variable

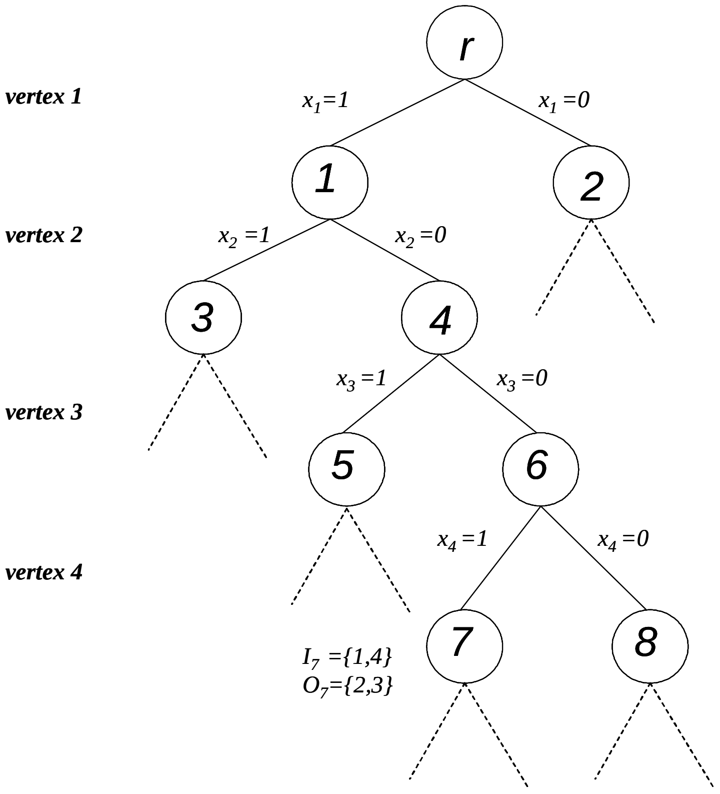

Figure 3: *The partial binary search tree associated with the graph in Figure 2.*

$x_i = 1$), while a backtrack operation removes the last selected vertex from the current partial solution (by setting the corresponding variable $x_i = 0$). At a generic node of the search tree, a forward move is performed if the current partial solution is not feasible and the lower bound computed at the node is strictly less than the incumbent feasible solution value. Otherwise, if the solution is feasible, a backtrack is carried out and the node is pruned. When a feasible solution is achieved, the node is pruned and the incumbent solution is could be updated. The algorithm stops when no further backtracking can be performed.

In particular, the lower bound associated with each decision node (used to carry out a pruning operation) is obtained by running the procedure modified-LB, which is described next in this section (see Table 5). A feasible initial solution is also associated with the root node; this is the solution provided by the genetic algorithm described in Section 4.1. To check the feasibility of the current solution we first verify whether property (6) is satisfied. If this is not the case, then the current solution is considered unfeasible. On the contrary, if property (6) is satisfied the current solution *might* be feasible and, to check that, we build the associated modified flow conservation system and check whether it admits a unique solution.
The pseudocode of the algorithm is given in Table 4 where the following notation is used:

$\hat{x}$ = vector of the current solution ($\hat{x}_i$ denotes the i-th component);

$\hat{z}$ = current solution value ($= \sum_{i=1}^{n} \hat{x}_i$);

x = vector of the best solution so far (x_i denotes the i-th component);

z = best solution value so far ($= \sum_{i=1}^{n} x_i$);

LB = value of the lower bound computed at the actual node;

4.2.1 The Modified-LB

The modified-LB algorithm, which is used to find a lower bound to the CSLP, is given in Table 5. We recall that the CSLP differs from the SLP in that a subset $I \subseteq N$ of vertices has already been designated to be selected, while a disjoint subset $O \subseteq N$ of vertices has been designated not to be part of the solution. The main idea of the algorithm consists of applying the LB procedure to each connected component of the graph $G' = G \backslash I$ resulting from G after the deletion of the vertices in I and after updating the degree of each vertex according to the set $Adj(I)$. In particular, for each (non- degenerate) not monitored connected component G'_i of G', the updated degree of each vertex $v \in N \backslash I$ counts the number of vertices in the component adjacent to v and not adjacent to any vertex in I of the component,

```
Step 0. (Initialization)
            $\hat{x} = 0$;
            $\hat{z} = 0$;
            $x$ and $z$ are returned by the genetic algorithm;
            $i = 1$.

Step 1. If $\hat{x}$ is feasible then:
                    - update $x$ and $z$;
                    - $x_i = 0$;
                    - Goto Step 3;
        If $LB < z$ then Goto Step 2 else $x_i = 0$.

Step 2. (Forward move)
            $i = i + 1$;
            $\hat{x}_i = 1$;
            Goto Step 1.

Step 3. (Backtrack move)
            $k = i$;
            while $k \geq 1$ do
                if $\hat{x}_k = 1$ then:
                    - $\hat{x}_k = 0$;
                    - $i = k$;
                    - Goto Step 2;
                $k = k - 1$;
            end while.

Step 4. (End of Algorithm)
            Return $x$ and $z$.
```

Table 4: Branch and Bound Algorithm: Overview.

```
Step 0. (Initialization)
        Let $G' = (N', A')$ be the subgraph of $G$ induced by the set of vertices $N' = N \setminus I$;
        Let $G'_i = (N'_i, A'_i)$, be the set of connected components of $G'$, $i = 1, 2, \ldots, k$;
        Set $M_i = \emptyset$ for each $i$.

Step 1. For each connected component $G'_i$ do
        1.1. Compute $\hat{\delta}(v) = |(Adj(v) \setminus Adj(I)) \cap N|$, for each $v \in N'_i$;
        1.2. Compute $\hat{b}_i = |B \cap N'_i| - |Adj(I) \cap N'_i|$;
        1.3. Sort the vertices in $N'_i \setminus O$ according to the decreasing values $\hat{\sigma}(v)$;
        1.4. While $\sum_{v \in M_i} \hat{\sigma}(v) < \hat{b}_i$ do
            1.4.1. Select the head vertex in $N'_i$, remove it from $N'_i$ and put it into $M_i$.

Step 2. Return $M = \cup_i M_i$.
```

Table 5: Modified-LB algorithm to compute a lower bound on the solution value of the CSLP.

that is, $\hat{\delta}(v) = |(Adj(v) \setminus Adj(I)) \cap N'|$. Moreover, the size of the set of OD vertices in each component is decreased by the total number of vertices in the component that are adjacent to vertices in I, that is, the quantity $|B \cap N'_i|$ decreases to $\hat{b}_i = |B \cap N'_i| - |Adj(I) \cap N'_i|$. All the vertices in the component, except those in O, are sorted in decreasing order according to the value $\hat{\sigma}(v)$, that is, equal to $\hat{\delta}(v) + 1$ if vertex v is an OD vertex and is equal to $\hat{\delta}(v)$ otherwise. Vertices are then chosen to be part of the selected set M_i until $\sum_{v \in M_i} \hat{\sigma}(v) \geq \hat{b}_i$. The final lower bound is given by summing up all the lower bounds obtained in each component, that is, $|M| = \sum_i |M_i|$.

5 Computational results on benchmark instances

To experimentally evaluate the algorithms, we test them on the benchmark instances described in [11] where two sets of graph classes were defined: complete grid graph scenarios and random grid graph scenarios. We also run our algorithm on the real traffic network of the city of Waynesboro, Virginia, USA. All the instances can be downloaded from the authors' website [1], [2]. All tests were performed on a workstation with Intel Core 2 Duo processor at 2.4Ghz and 3GB of RAM. All the procedures have been coded in C++. Library Lapack was used to compute the rank of a matrix. The mathematical model was solved using the AMPL language with IBM ILOG CPLEX 12.1. A time limit of one hour was set for the branch and bound algorithm, and a time limit of 10 minutes was set to solve the mathematical model.

Complete Grid Graphs Scenarios

There are complete grid graphs of dimension 6×6, 8×8, 10×10, 15×15 and 20×20. The distribution of the OD vertices was generated from a uniform stochastic variable by considering the total number of OD vertices to be equal to $|B| = n/2$ and $|B| = n/4$, where n is the number of vertices in the grid. The split ratios associated with the outgoing arcs of each vertex v are equal to $1/\delta(v)$. For each grid, and a given number of OD vertices, the reported results are the average size of the returned set M on 10 runs, for a total of 120 instances.

Modified Grid Graphs Scenarios

Starting from complete grid graphs, new random graphs with different arc densities were generated. In particular, given a complete grid graph $G = (N, A)$, the value π is the probability that each arc belongs to the new graph. If the resulting graph is not connected, a simple procedure is performed to add arcs between vertices of different connected components. Five different complete grid graphs were considered whose size are, 6×6, 8×8, 10×10, 15×15 and 20×20. For each modified grid obtained with $\pi \in \{0.01, 0.03, 0.04\}$, the distribution of the OD vertices was generated from a uniform stochastic variable by considering the total number of OD vertices to be equal to $|B| = n/2$ and $|B| = n/4$, where n is the number of vertices in the grid. The split ratios associated with the outgoing arcs of each vertex v is equal to $1/\delta(v)$. For each grid, each possible value of π and a fixed the number of OD vertices, the reported results are the average size of the set M on 10 runs, for a total of 360 instances.

Table 6 and Table 7 show the results obtained on the complete grid graph scenarios and on the modified grid graph scenarios, respectively. All the computational times are expressed in seconds. Column n reports number of vertices, column b reports number of OD vertices, and column π contains the probability values for arc generation. Columns *BB Sol.* and *BB Time* contain the solution and the computational time for our Branch and Bound algorithm. Columns *Genetic Sol.* and *Genetic Time* contain the solution and the computational time for the genetic algorithm. The lower bound provided by the LB procedure is shown in column LB, the corresponding computational time is reported in column *LB Time.* Finally, the last six columns of each table show the lower bounds obtained by solving our mathematical model. In particular, columns *Integer Sol.* and *Integer Time* contain the optimum solution of the model and the corresponding computational time required by the solver. We outline that the optimum solution of the mathematical formulation provides a lower bound on the optimum solution value of the SLP. Indeed, the formulation seeks a subset of vertices M of minimum size such that a necessary condition for M to be a monitoring set is satisfied. However, since the condition is not sufficient, the resulting set could not be feasible for the original problem. We also obtained two additional lower bounds by considering two relaxations of the model: the mixed integer relaxation obtained by relaxing the integer requirement on variables y (that is, constraints (15)) and the linear relaxation obtained by relaxing the integer requirement both on variables y and z (that is, constraints (14) and (15)). The corresponding results and computational time are reported in columns *Mixed Sol.* and *Mixed Time* for the mixed integer relaxation, and in columns *Relaxed Sol.* and *Relaxed Time* for the linear relaxation. The upper part of both the tables shows the results obtained by setting the number of OD vertices b equal to $\frac{n}{2}$, the lower part of the tables contains the results for values of b equal to $\frac{n}{4}$. The reported results are the average value of ten instances. When the symbol '*' appears in the *BB Sol.* column, it means that in at least one instance of the scenario the time limit of 3600 seconds was reached. When the symbol '*' appears in the *Integer Sol.* or *Mixed Sol.* column, it means that in at least one instance of the scenario the time limit of 360 seconds was reached.

The branch and bound approach has negligible computational times on small instances (i.e. $n = 36, 64$), both for the regular grids and for the modified grids. It reaches the time limit on all the 10 instances of a scenario when $n = 225, 400$ for both the class of grids, and also when n=100 and the number of OD vertices is equal to 50. This reveals that both the size of the graph and the number of OD vertices determine the difficulty of the scenario, as was expected.
Our genetic algorithm always finds the optimum solution when this was known. Moreover, the solution returned by the BB when time limit is reached is the same solution returned by the genetic algorithm. Hence, all the time spent by the BB approach is used to test optimality of the initial solution provided by the genetic algorithm. Computational times of the genetic algorithm are proportional to the number of vertices. Hence, this approach is slow compared to the exact branch and bound on the small instances, while it outperforms the exact approach on more difficult and larger scenarios, but its computational time tends to increase rapidly with the size of the instance. The main drawback of both approaches lies in the feasibility test of a solution. Indeed, to test whether a given set of vertices is a monitoring set, the rank of a square matrix has to be computed by means of the Lapack library and the computational time of the algorithms, given that such a feasibility test is carried out several times for a huge matrix, increases very fast as the number of vertices and the number of OD vertices increase. This result was somehow expected and shows the importance of defining necessary and sufficient conditions to test for feasibility of a given subset of vertices that make use of the combinatorial nature of the problem, and the importance of the underlying structure network, so that more efficient algorithms can be designed.

Let us compare the quality of the lower bounds returned by the three mathematical models and by the LB algorithm. First of all, observe that the computational times for the LB algorithm are always negligible compared to the computational times required by the solver to solve any of the three mathematical models, both for regular grids and for modified grids. Analyzing the results on regular grids more in detail (Table 6), we observe that the solution value returned by the LB algorithm and the optimum solution of the mathematical model are the same for the complete grid graph scenarios, and such a solution has a value equal to the solution returned by the genetic algorithm on instances with $n = 36$. This explains the negligible time of the BB algorithm on these scenarios: in most of these cases, it does not even generate any node in the search tree, since it finds the optimum solution on the root node. The mixed and linear relaxations of the models require less computational time. The solution returned by the Mixed relaxation is always equal to the optimum solution of the model (except for two instances with $n = 400$ and $b = 200$ where the time limit of 360 seconds was reached).

Let us analyze the quality of the lower bounds on the modified grid graph. Consider Table 8 where we reported the percentage gaps of the solution values among different approaches. In particular, each column $A_1 \to A_2$ is computed as $\frac{A_1 - A_2}{A_1}$: if the gap is negative then the solution value returned by A_1 is less that the solution value returned by A_2; otherwise, it is greater. The smaller the absolute value of the gap, the more the two solutions are similar. When the mathematical model (either when solved to optimality or when the mixed or linear relaxations are considered) reached the time limit, the gap cannot be computed and the symbol NA is reported.

We can observe that the lower bounds returned by the LB algorithm and the lower bounds returned by the solution of the mixed relaxation of the model are similar (see column LB→Mixed). Analyzing the values on column Integer→LB, we can see the lower bound of LB procedure is obviously always smaller than the optimum solution of the mathematical model. The difference is smaller on those scenarios with $\pi = 0.01$, that is, those scenarios that are more similar to the complete grid scenarios, as was expected. The maximum gap is equal to 33% and it is reached when $n = 100$ and $\pi = 0.04$. We can also observe that in those scenarios with the same number of vertices, the lower bound is worse when the number of OD vertices in greater, as was expected.

6 Computational results on a real worls network

We also ran our algorithm on the real traffic network of the city of Waynesboro, Virginia, USA. The entire network was imported from the OpenStreetMap website [3]. The network has 1306 vertices and 3532 arcs. We randomly selected 100 vertices as OD vertices. Figure 4 shows the network where the black vertices are OD vertices. The genetic algorithm was stopped after a time limit of 1 hour and returned a feasible solution equal to 40. The Branch and Bound algorithm was not able to improve such a solution after three hours of computation. The LB algorithm returned a lower bound equal to 20 in 0.0 seconds. The lower bound returned by the mathematical model was equal to 20 and the required computational time was equal to 4.65 seconds. The mixed and the linear relaxation returned solutions equal to 20 and 19.8 in 0.2 and 0.0 seconds, respectively.

7 Conclusions

In this paper we extend the study of a known NP-complete optimization problem that finds application in traffic management and control: the Sensor Location Problem. The existing contributions in the literature about the problem consist of the study of the computational complexity and the definition of greedy heuristics to get lower and upper bounds. Our contributions fill some gaps present in the literature about the problem: (i) we provide a mathematical formulation for the problem (missing in the literature until now), (ii) we developed and tested an exact branch and bound approach to find the optimum solution of the problem (iii) we developed and tested a genetic approach to find good quality solutions. Our computational results were carried out on existing benchmark instances and show that the proposed metaheuristic is very efficient for solving the problem on medium-large instances. Our computational study revealed that the *weak* point of any solution algorithm is the feasibility test, which currently requires the computation of the rank of a matrix. Since such a feasibility test is used several times in any given algorithm, it is of great relevance, for the designing of more efficient algorithms, to better analyze the combinatorial structure of the problem and define necessary and sufficient conditions for testing feasibility of a given subset of vertices. This is one of the issues our further study of the problem is focusing on. We are also studying interesting variants of the problems. In particular, we would like to extend the results considering the weighted version of the problem, that is, when there is a different cost associated with the location of a sensor both on directed graphs and on bi-directed graphs.

Acknowledgements

This work was partially supported by MIUR Project n. 2007J494P3_004, year 2007.
The authors would like to thank the two anonymous referees, whose comments and suggestions helped improve the quality of the paper.. The list of authors on the first page was arbitrarily sequenced in alphabetical order. Each author was an equal contributor to the final product.

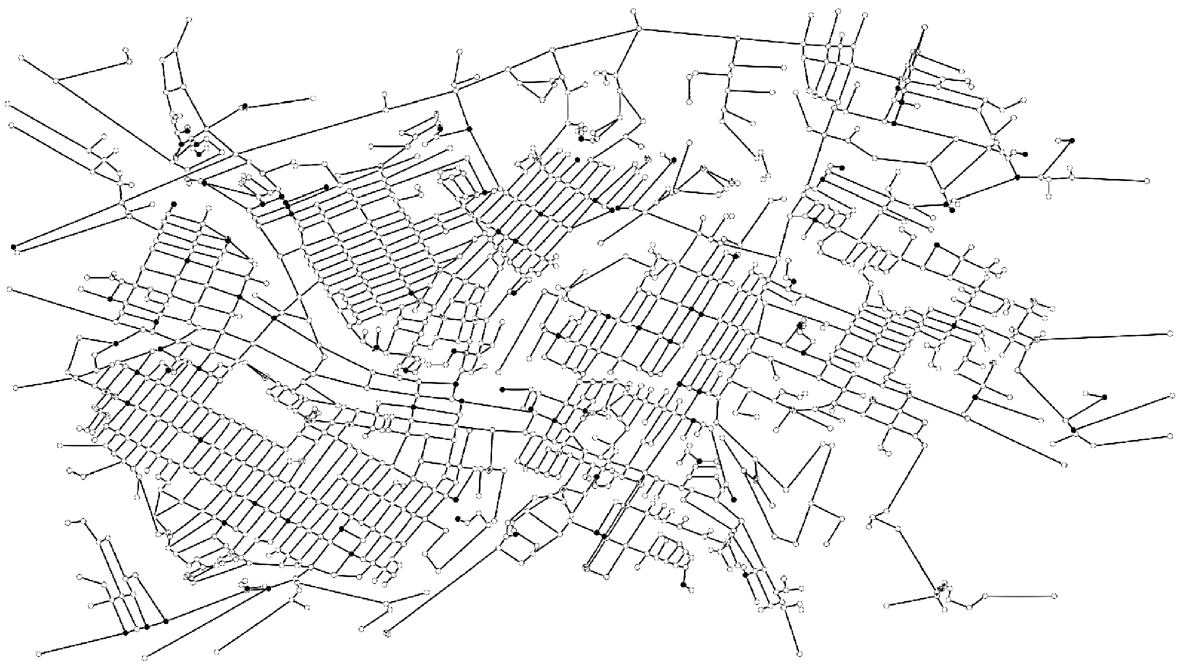

Figure 4: *Waynesboro Network, Virginia, USA.*

Table 6: *Results Comparison for Complete Grids*

n	b	**BB Sol.**	**BB Time**	**Genetic Sol.**	**Genetic Time**	**LB Sol.**	**LB Time**	**Integer Sol.**	**Integer Time**	**Mixed Sol.**	**Mixed Time**	**Relaxed Sol.**	**Relaxed Time**
36	18	4.00	0.00	4.00	36.00	4.00	0.00	4.00	0.10	4.00	0.01	3.60	0.01
64	32	7.00	2.63	7.10	64.01	7.00	0.00	7.00	0.24	7.00	0.07	6.40	0.01
100	50	11.3*	3600.00	11.30	100.06	10.00	0.00	10.00	0.91	10.00	1.51	10.00	0.03
225	112	27.2*	3600.01	27.20	228.15	23.00	0.00	23.00	10.50	23.00	0.81	22.40	0.11
400	200	51.8*	3600.01	51.80	440.18	40.00	0.00	40.00	34.21	40.2*	167.40	40.00	0.36
36	9	2.00	0.00	2.00	36.00	2.00	0.00	2.00	0.03	2.00	0.02	1.80	0.01
64	16	4.00	0.00	4.00	64.08	4.00	0.00	4.00	0.14	4.00	0.07	3.20	0.01
100	25	5.00	0.33	5.20	100.07	5.00	0.00	5.00	0.69	5.00	0.34	5.00	0.02
225	56	13*	3600.01	13.00	234.52	12.00	0.00	12.00	2.69	12.00	0.64	11.20	0.10
400	100	24.9*	3601.87	24.90	840.65	20.00	0.00	20.00	16.98	20.00	19.80	20.00	0.29

Appendix

A subset M of vertices is a monitoring set if and only if the associated modified flow conservation system has a unique solution, that is, the coefficient matrix E_M, of such a system has full rank. In this section, we describe the procedure for building the modified flow conservation system, and therefore matrix E_M, associated with a subset M of vertices. The description and definitions given here are the same as those given in [21] and [22].

Table 7: *Results Comparison for Modified Grids*

n	π	b	**BB Sol.**	**BB Time**	**Genetic Sol.**	**Genetic Time**	**LB Sol.**	**LB Time**	**Integer Sol.**	**Integer Time**	**Mixed Sol.**	**Mixed Time**	**Relaxed Sol.**	**Relaxed Time**
36	0.01	18	5.00	0.05	5.00	36.00	4.10	0.00	4.60	0.17	4.40	0.02	3.81	0.01
36	0.03	18	6.00	0.02	6.00	36.00	4.80	0.00	6.00	0.12	4.80	0.02	4.02	0.01
36	0.04	18	6.20	0.06	6.20	36.00	4.30	0.00	6.10	0.13	4.40	0.01	3.89	0.01
64	0.01	32	8.60	202.55	8.60	64.00	7.00	0.00	7.70	2.19	7.10	0.04	6.54	0.01
64	0.03	32	11.0*	1487.67	11.00	64.00	7.50	0.00	10.70	0.89	7.60	0.03	7.04	0.01
64	0.04	32	11.6*	2115.34	11.60	64.00	7.60	0.00	11.10	1.02	7.90	0.03	7.26	0.01
100	0.01	50	12.6*	3241.48	12.60	100.03	10.00	0.00	11.20	2.32	10.90	0.18	9.99	0.02
100	0.03	50	17.2*	3600.00	17.20	100.02	11.50	0.00	16.30	4.32	11.80	0.05	11.05	0.01
100	0.04	50	18.0*	3600.00	18.00	100.02	11.40	0.00	17.00	4.31	11.50	0.06	10.81	0.01
225	0.01	112	32.2*	3600.02	32.20	230.49	23.00	0.00	24.60	25.17	23.00	0.38	22.40	0.07
225	0.03	112	40.9*	3600.01	40.90	225.25	25.40	0.00	36.3*	336.83	25.20	0.12	24.66	0.04
225	0.04	112	44.7*	3600.00	44.70	224.34	25.10	0.00	38.6*	335.14	24.70	0.15	24.05	0.03
400	0.01	200	63.1*	3600.01	63.10	419.54	40.00	0.00	43.1*	168.27	40.9*	54.56	39.98	0.21
400	0.03	200	96.1*	3600.01	96.10	400.00	44.60	0.00	63.8*	361.19	44.40	0.36	43.68	0.08
400	0.04	200	98.6*	3600.01	99.40	400.00	43.70	0.00	69.2*	360.88	43.90	0.39	43.34	0.07
36	0.01	9	2.90	0.00	2.90	36.00	2.50	0.00	2.60	0.05	2.00	0.01	1.78	0.01
36	0.03	9	3.10	0.00	3.10	36.00	2.30	0.00	2.90	0.08	2.20	0.01	1.80	0.01
36	0.04	9	3.10	0.00	3.10	36.00	2.20	0.00	3.10	0.07	2.20	0.01	1.73	0.00
64	0.01	16	4.00	0.00	4.00	64.01	4.00	0.00	4.00	0.06	4.00	0.03	3.20	0.01
64	0.03	16	5.70	0.43	5.70	64.00	3.90	0.00	5.20	0.51	3.80	0.03	3.04	0.01
64	0.04	16	5.90	1.57	5.90	64.00	3.80	0.00	5.60	0.39	4.00	0.03	3.20	0.01
100	0.01	25	6.00	15.98	6.10	100.09	5.10	0.00	5.60	0.29	5.50	0.23	4.98	0.02
100	0.03	25	8.5*	1387.53	8.60	100.05	5.80	0.00	7.80	1.20	5.50	0.04	4.86	0.01
100	0.04	25	8.7*	1489.17	8.70	100.02	5.50	0.00	8.20	1.16	5.30	0.04	4.75	0.01
225	0.01	56	15.6*	3614.36	15.60	232.19	12.00	0.00	12.00	0.96	12.00	0.23	11.20	0.06
225	0.03	56	21.6*	3600.04	21.60	224.89	12.70	0.00	16.7*	78.83	11.40	0.09	10.93	0.03
225	0.04	56	22.8*	3600.00	22.80	224.43	12.40	0.00	18.1*	127.65	11.20	0.11	10.44	0.03
400	0.01	100	31.0*	3739.66	31.00	439.98	20.00	0.00	20.6*	43.68	20.5*	61.48	19.98	0.16
400	0.03	100	46.9*	3600.01	46.90	401.94	22.20	0.00	30.0*	283.20	20.00	0.26	19.39	0.07
400	0.04	100	49.4*	3600.01	49.40	400.00	20.70	0.00	32.7*	330.38	19.10	0.22	18.54	0.06

In particular, let us consider a bi-directed graph $G = (N, A)$ with N the set of n vertices and A the set of m arcs, a subset $B \subseteq N$ of OD vertices, the set of turning ratios $p_{v,w}$ associated with each arc $(v, w) \in A$ and finally a subset M of vertices. The procedure described next first builds a matrix E associated with G, B and the turning ratios $p_{v,w}$ (Steps 1-5), and then it opportunely eliminates rows and columns from E to obtain the submatrix E_M (Steps 6-8).

Procedure for building matrix E_M

Step 1 Select for each vertex $i \in N$ an outgoing arc $e_i = (i, w)$ that will be referred to as the principal arc of vertex i;

Step 2 For each arc $(i, j) \in A$ compute the turning factor α_{ij} as the ratio between the turning ratio of the arc (i, j) and the turning ratio of the principal arc of vertex i, that is, $\alpha_{i,j} = \frac{p_{i,j}}{p_{i,w}}$;

Table 8: *Percentage Gap Comparison for Modified Grids*

n	π	b	**Integer →LB**	**Integer→Mixed**	**Integer →Relaxed**	**LB→Mixed**	**LB→Relaxed**
36	0.01	18	10%	4%	16%	-8%	7%
36	0.03	18	20%	20%	33%	-1%	16%
36	0.04	18	29%	27%	36%	-3%	9%
64	0.01	32	9%	8%	15%	-1%	7%
64	0.03	32	30%	29%	34%	-2%	6%
64	0.04	32	31%	28%	34%	-4%	4%
100	0.01	50	11%	3%	11%	-9%	0%
100	0.03	50	29%	27%	32%	-3%	4%
100	0.04	50	33%	32%	36%	-1%	5%
225	0.01	112	6%	6%	9%	0%	3%
225	0.03	112	NA	NA	NA	1%	3%
225	0.04	112	NA	NA	NA	2%	4%
400	0.01	200	NA	NA	NA	NA	0%
400	0.03	200	NA	NA	NA	0%	2%
400	0.04	200	NA	NA	NA	0%	1%
36	0.01	9	3%	20%	29%	17%	26%
36	0.03	9	20%	23%	37%	3%	20%
36	0.04	9	28%	28%	44%	0%	20%
64	0.01	16	0%	0%	20%	0%	20%
64	0.03	16	25%	27%	41%	3%	22%
64	0.04	16	32%	28%	42%	-7%	15%
100	0.01	25	8%	2%	10%	-8%	2%
100	0.03	25	25%	29%	37%	5%	16%
100	0.04	25	33%	35%	42%	3%	13%
225	0.01	56	0%	0%	7%	0%	7%
225	0.03	56	NA	NA	NA	10%	14%
225	0.04	56	NA	NA	NA	9%	16%
400	0.01	100	NA	NA	NA	-3%	0%
400	0.03	100	NA	NA	NA	10%	13%
400	0.04	100	NA	NA	NA	8%	10%

Step 3 Build a square matrix $H = \{h_{ij}\}$ with n rows and n columns as follows:

$$h_{ij} = \begin{cases} \alpha_{ij} & \text{if vertex } i \text{ and vertex } j \text{ are connected} \\ -\sum_{w \in A^-(i)} \alpha_{ij} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Note that each row and each column of matrix H are associated with a vertex of the graph.

Step 4 Build the matrix $C = \{c_{ij}\}$ with n rows and $|B|$ columns, where each row is associated with a vertex and each column is associated with an OD vertex in the graph. The generic column of matrix C associated with the OD vertex i is such that all its elements are equal to 0 except the element corresponding to row i, which is equal to 1.

Step 5 Let $E = [H|C]$ be the adjunct matrix obtained by appending the columns of matrix C to the columns of matrix H.

Step 6 For each vertex $i \in M$ do the following:

- delete from E the row corresponding to vertex i;

- delete from E the column of submatrix H corresponding to vertex i;

- if vertex $i \in B$, delete from E the column of submatrix C corresponding to the OD vertex i;

Step 7 For each vertex $j \in Adj(M)$ delete from E the column of submatrix H corresponding to vertex j;

Step 8 Let E_M be the resulting matrix.

References

[1] http://www.dipmat.unisa.it/people/cerulli/www/.

[2] http://www.dmi.unisa.it/people/gentili/www/publicationm.htm.

[3] http://www.openstreetmap.org/.

[4] L. Bianco, G. Confessore, and M. Gentili. Combinatorial aspects of the sensor location problem. *Annals of Operations Research*, 144:201–234, 2006.

[5] L. Bianco, G. Confessore, and P. Reverberi. A network based model for traffic sensor location with implication in o-d matrix estimates. *Transportation Science*, 35:50–60, 2001.

[6] E. Castillo, A.J. Conejo, J.M. Menéndez, and P. Jiménez. The observability problem in traffic network models. *Computer-Aided Civil and Infrastructure Engineering*, 23:208–222, 2008.

[7] E. Castillo, A.J. Conejo, R.E. Pruneda, and C. Solares. Observability in linear systems of equations and inequalities: Applications. *Computers and Operations Research*, 34:1708–1720, 2007.

[8] E. Castillo, P. Jiménez, J.M. Menéndez, and A.J. Conejo. The observability problem in traffic models: Algebraic and topological methods. *IEEE Transactions on Intelligent Transportation Systems*, 9:275–287, 2008.

[9] C. Cerrone, R. Cerulli, and M. Gentili. Vehicle-id sensor location for route flow recognition: Models and algorithms. *Submitted to Transportation Research Part B.*

[10] P. Chootinan, A. Chen, and H. Yang. A bi-objective traffic location problem for origin -destination trip table estimation. *Transportmetrica*, 1:65–80, 2005.

[11] G. Confessore, P. Dell'Olmo, and M. Gentili. Experimental evaluation of approximation and heuristic algorithms for the dominating paths problem. *Computers and Operations Research*, 32:2383–2405, 2005.

[12] A. Elhert, M.G.H. Bell, and S. Grosso. The optimization of traffic count locations in road networks. *Transportation Research Part B*, 40:460–479, 2006.

[13] M. Gendreau, G. Laporte, and I. Parent. Heuristics for the location of inspection stations on a network. *Naval Research Logistics*, 472:287–303, 2000.

[14] M. Gentili and P. Mirchandani. Survey of models to locate sensors to estimate traffic flows. *Transportation Research Record: Journal of the Transportation Research Board*, 2243:108–116, 2011.

[15] M. Gentili and P. Mirchandani. Locating sensors on traffic networks: Models, challenges and research opportunities. *Transportation Research Part C: Emerging Technologies*, 24:227255, 2012.

[16] M. Gentili and P.B. Mirchandani. Location of active sensors on traffic network. *Annals of Operations Research*, 136:229–257, 2005.

[17] M.J. Hodgson. A flow-capturing location-allocation model. *Geographical Analysis*, 22:270–279, 1990.

[18] H.Yang, C. Yang, and L. Gan. Models and algorithms for the screen line-based traffic counting location problem. *Computers and Operations Research*, 33:836–858, 2006.

[19] H.J. Kim, H. Chung, and S.Y. Chung. Selection of the optimal traffic counting locations for estimating origin-destination trip matrix. *Journal of Eastern Asian Society for Transportation Studies*, 5:1353 – 1365, 2003.

[20] W.H.K. Lam and H.P. Lo. Accuracy of o-d estimates from traffic counts. *Traffic Engineering and Control*, 31:435–447, 1990.

[21] D. Morrison and S. Martonosi. Where are all the cars? characteristics of optimal solutions to the sensor location problem. In *INFORMS Annual Meeting 2008, Washington D.C. October 12-15, 2008.*

[22] D. Morrison, S. Martonosi, and K. Tucker. Characteristics of optimal solutions to the sensor location problem. *Thesis. Harvey Mudd College, Department of Mathematics*, 2008.

[23] H. Yang and J. Zhou. Optimal traffic counting locations for origin-destination matrix estimation. *Transportation Research Part B*, 32:109–126, 1998.